EXPRESS MAIL NO. EL652176450US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Siani Lynne PEARSON, et al.	Re: Claim To Priority
U.S. Appln. No.: not yet assigned	Group: not yet assigned
U.S. Filing Date: concurrently herewith	Examiner: not yet assigned
International Application No: PCT/GB00/03095	
International Filing Date: 11 August 2000	Our Ref.: B-4488PCT 619500-7
For: "ENFORCING RESTRICTIONS ON THE USE OF STORED DATA"	Date: February 5, 2002

Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Attn: United States Designated/Elected Office (DO/EO/US)

35 U.S.C. 119 CLAIM TO PRIORITY

Sir:

Prior PCT International Application No. PCT/GB00/03095, designating the U.S., claims foreign priority as follows:

COUNTRY	FILING DATE	SERIAL NUMBER
(EP) European Patent Office	13 August 1999	99306415.3
Great Britain	25 September 1999	9922669.8

A certified copy has been filed in prior PCT International Patent Application No. PCT/GB00/03095.

Applicants hereby confirm that this claim for priority applies to the above-identified U.S. International Stage Application.

Respectfully submitted,

Richard P. Berg
Reg. No. 28,145
Attorney for Applicant
LADAS & PARRY
5670 Wilshire Boulevard #2100
Los Angeles, California 90036
(323) 934-2300

PCT/GB 0 0 / 0 3 0 9 5

The Patent Office

GB 00/03095

INVESTOR IN PEOPLE

09/049213

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

REC'D 0 4 SEP 2000
WIPO PCT

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed

Dated 22 NOV 1999

Patents Form 1/77

Patents Act 1977
(Rule 16)

The Patent Office

27SEP99 E479395-1 001463
P01/7700 0.00 - 9922669.8

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference — 309901.41 GB

2. Patent application number *(The Patent Office will fill in this part)* — **9922669.8**

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, California 94304 USA

Patents ADP number *(if you know it)* 00496588004

If the applicant is a corporate body, give the country/state of its incorporation — Delaware USA

4. Title of the invention

ENFORCING RESTRICTIONS ON THE USE OF STORED DATA USING A COMBINATION OF MULTIPLE TRUSTED DEVICES

5. Name of your agent *(if you have one)*

LAWMAN Matthew John Mitchell

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

Hewlett-Packard Limited
IP Section
Filton Road
Stoke Gifford
BRISTOL BS34 8QZ

THE PATENT OFFICE
25 SEP 1999
RECEIVED BY POST

Patents ADP number *(if you know it)* 07337009001

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

Yes

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	
Description	14
Claim(s)	
Abstract	
Drawing(s)	6 + 6

10. If you are also filing any of the following, state how many against each item.

Priority documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	
Request for preliminary examination and search *(Patents Form 9/77)*	X
Request for substantive examination *(Patents Form 10/77)*	
Any other documents *(please specify)*	Annex 'A' - unpublished European patent application No 99301100.6

11. I/We request the grant of a patent on the basis of this application.

Signature Matthew Lawman Date 24/09/99

12. Name and daytime telephone number of person to contact in the United Kingdom Katerina Nonmeots-Nomm 0117 312 9947

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.

b) Write your answers in capital letters using black ink or you may type them.

c) If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.

d) If you have answered 'Yes' Patents Form 7/77 will need to be filed.

e) Once you have filled in the form you must remember to sign and date it.

f) For details of the fee and ways to pay please contact the Patent Office.

Enforcing restrictions on the use of stored data using a combination of multiple trusted devices

(HP Ref: 3099014-1)

This invention provides a method for allowing mobile users to have universal data access on trusted computer platforms by enforcing restrictions on the use of stored data via a user's licence associated with the data together with software that checks the validity of operations carried out on the data. The user's licence can be stored on or issued with a portable trusted device such as a smart card, downloaded together with the data, or sent separately from the data. There is an option to perform integrity checks on the data to ensure that the data has not been modified since installation. Hence, unauthorised operations on data such as copying can be prevented, together with modification of data or its associated licence on the same platform, while users can benefit from a hot-desking model of data access.

In this document, 'data' signifies anything that can be formatted digitally, such as images, software and streaming media.

Applicant's co-pending patent application EP 99301100.6 "Trusted Computing Platform", the entire contents of which are hereby incorporated herein by reference (and which accompanies this application as Annex A), describes the use of a trusted component (TC) to enable verification of the integrity of a computer platform by the reliable measurement and reliable reporting of integrity metrics. This enables the verification of the integrity of a platform by either a local user or a remote entity. That prior patent application describes a general method of reporting integrity metrics and verifying the correctness of the integrity of a platform by comparing reported values of metrics with proper values of metrics.

The present invention applies to software wrappers or other types of data licence used to protect and qualify the operations that may be performed upon data, such as copying, modification, or execution. Data wrappers, or cryptographic containers, are commonly used within software-only and hybrid methods of data protection, but are not at present a very secure method of protection because they are vulnerable to alteration and removal, even if an integrity check is contained within the wrapper.

The present invention uses two TCs or "trusted modules": the first is a portable TC such as a smart card, and the second is part of a computer platform. These are used in conjunction with software, preferably running within the TC, to ensure that data can only be used by the owner of the portable TC in ways specified by the developer, and furthermore can include a check that the associated access profile or other type of wrapper has not been

altered or deleted after the data (together with access profile or wrapper) have been s on the trusted client platform.

A system has already been proposed in "Persistent Access Control to Prevent Piracy of Digital Information" Paul B. Schneck, Proceedings of the IEEE, Volume 87, No. 7, July 1999, PP1239-1250, which uses access control software to check licensing information before access to data. That system, however, only considered the case where a generic licence was operable for all users. There are many situations where a platform is shared by users who have different software permissions. The present invention allows individual users to pay for data access that only they will benefit from. It allows users to be able to capture such a licence on a tamper-resistant portable device that they can carry around with them, and use on any trusted platform, no matter where its location. Alternatively, individual licences held on trusted platforms can be customised to refer to a secure ID of the portable module. If the data itself if also captured on the portable device, it is not necessary for the data to be installed on the client machine. Optionally, the data can be downloaded as required, if not already installed.

The present invention differs from Schneck's system in that it uses a client system where access checks are made according to the user identity corresponding to a removable TC such as a smart card, but the checks themselves are made using the access control software mounted on the client PC. Furthermore: (a) a licence can be associated with each end-user, instead of or as well as with the PC TC, which is necessary for certain types of access control; (b) it is not essential (although it is preferable) that the data is encrypted (c); to prevent modification of the licence (cf. access profile), a digest is stored in the TC upon loading and consulted before data access; (d) the access control code is protected at BIS (BOOT Integrity Service) and preferably runs within the TC, or else, there is a dedicated communications path between the code and TC that is inaccessible to other parts of the computer platform; (e) logs are made within the TC; and (f) the licence can have a more proactive role.

A common technique in the field of licence protection is to use a software wrapper to encode information relating to licensing and other protection measures. However, the data wrapper is a prime target for hackers since it may contain a profile defined by the data's developer that governs the way in which the data may be executed, or other sensitive information which should not be altered. Authentication, encryption and integrity checks may be used to protect the wrapper from being modified en route to its being downloaded and stored onto the client platform. However, there is a major risk that it could be modified or deleted by a malicious entity, or by accident, once the data and associated wrapper are

stored (for example, on a hard disk) within the client platform. Once modified, the data could then be used on the client platform in a way that is outside the scope of the profile defined in the original, unmodified wrapper. Access control mechanisms cannot provide a complete solution to this problem because they can be bypassed and, moreover, they focus on controls specified by the user's administrator rather than the developer of the data.

The motivation for this particular invention is that more complex models of data usage dictate greater flexibility, which can only be brought about by using multiple TCs in the client platform. In particular, hot-desking in an office environment or accessing information or services in from a shared terminal in a public place such as an airport can be modelled by having a TC in a shared client machine, and each user being issued with at least one portable TC that identifies this user.

In overview, a preferred embodiment of the present invention uses a TC, or "trusted module" of a computer platform (that is possibly shared by several users) in conjunction with software, preferably running within the TC, that ensures that restriction on the usage by each individual end-user of stored data specified by the developer must be adhered to, that the different end-users can have different access profiles and in addition that data cannot be used on the platform if the data or associated wrapper or licence has been modified since the initial download onto the platform.

More specifically, the invention is that the host CPU requests a policy check before acting upon data, by sending the name of the target data plus the intended operation to a TC. The TC checks the ID of the user that is logged in (via the ID of the portable TC), and the restrictions corresponding to this current end-user that are associated with the target data. Those restrictions could be on who may access the data, on the number of times the data can be used, the operations which may not be carried out, and so on, or the restrictions might have deliberately been loaded as 'NULL'. The TC checks the proposed usage with the restrictions. If no appropriate permission of found, the TC checks for a licence on the portable TC (e.g. a smart card) and for valid permission for the data usage within this. The TC then replies to the CPU with or without the access permission, as appropriate. The CPU is not able to carry out certain operations on target data such as copy, edit, add section, replace section, execute, delete, print, open, scan, rename, move location, send to or read without obtaining the appropriate permission from the TC in such a manner. Preferably, the integrity of the target data and restrictions is checked before the operation is carried out to ensure that they have not been illegally or accidentally modified on the platform. Alternatively, the checking can be carried out on the portable TC itself.

More formally, in accordance with a first aspect of the present invention, th provided a computer platform having: a trusted module which is resistant to internal tampering and which stores a third party's public key certificate; means of storing operation protection code comprising at least one of: a secure operator (which is preferably generic) for checking whether the platform or a user thereof has permission from the developer to perform an operation on particular data; and a data protector for checking data upon installation and ensuring that data cannot be operated upon on the platform if the data or associated access profile has been modified since the initial download onto the platform; and means of storing a hashed version of the operation protection code signed with the third party's private key; wherein the computer platform is programmed so that, upon booting of the platform: the operation protection code is integrity checked with reference to the signed version and the public key certificate; and if the integrity check fails, the operation protection code is prevented from being loaded. If the integrity check fails, it may be arranged that the complete platform integrity fails.

The trusted module or component, as described in EP 99301100.6, is preferably immune to unauthorised modification or inspection of internal data. It is physical to prevent forgery, tamper-resistant to prevent counterfeiting, and preferably has crypto functions to securely communicate at a distance. Methods of building trusted modules are, per se, well known to those skilled in the art. The trusted module may use cryptographic methods to give itself a cryptographic identity and to provide authenticity, integrity, confidentiality, guard against replay attacks, make digital signatures, and use digital certificates as required. These and other crypto methods and their initialisation are well known to those skilled in the art of security.

Preferably, an additional component is the access profile associated with each piece of application software or data, which:

specifies the data which are to be protected;

specifies the type of operations that the developer wishes to be carried out upon that particular software or data;

optionally specifies any other particular information to be checked for in carrying out certain operations, such as a particular TC ID or a secret which is to be checked for in the TC or current signed-on smart card; and/or

optionally runs, together with the data, within a TC or smart card (suitably segmented).

The access profile can be thought of as a form of licence or cryptographic container associated with the data.

The platform includes a further, removable, trusted module (such as a smart card) that includes a user identity. Optionally, this may contain a user licence associated with a piece of data, which:

specifies the data which are to be protected; and/or

specifies the type of operations that the owner of the module is licensed to carry out on this data, together with any restrictions on this usage.

Upon sign-on, the removable module and the PC TC mutually authenticate and the TC stores the identifier of the removable module. Before protected data can be used, the secure operator or access profile associated with the data (depending upon the particular model used) need to give permission to the OS to carry out the particular operation. Upon checking the restrictions relating to the data, the secure operator or access profile is operable to perform the restrictions check with reference to the user identity. If the licence is stored in the smart card, the secure operator needs to retrieve the licence into a store held on the TC PC that it can consult in future, or else consult the smart card each time to find out the details of the licence. This user licence may be updated as a result of the data access: for instance, if an operation permission is qualified by being for a fixed number of uses.

The developer can issue (user) licences on smart cards, which would then be sent out to end-users after registration, or the licence can be downloaded electronically either to the smart card or to the PC TC. Data can be downloaded at the same time, or transferred separately, possibly by non-electronic means such as CD-ROM.

The procedures by which the system operates depend very much upon the particular trusted relationships in force between the developer, client PC and a trusted portable module such as a smart card. In general, the TC must be registered with the data-provider in order that the data can be sent to the TC (or analogously, to the smart card, if the data is to be sent to the smart card). The smart card must also be registered with the licence-provider (very probably the same entity as the data-provider), in order that the user ID of the smart card can be incorporated into the licence before it is issued to the TC. This would be a suitable model for circumstances such as when given users share a PC, in an office environment for example. However, in scenarios where the users of a client machine will not be known in advance, such as where machines are available in public places such as airports, this approach is not possible. Instead, the licence needs to be customised to the user ID of the smart card, and given to the end-user either by a new smart card being issued by the licence-provider, or by this information being downloaded into one already held by the end-user. The licence will contain a reference to the name and version of the software, if appropriate, and the ways in which that software can be used by the end-user. When the

data is installed into such a public shared trusted terminal, optionally a different access profile can be installed that can specify default restrictions upon the data installed upon it, or overriding restrictions, or a combination of both. For example, copying of a document could be forbidden unless an end-user specifically had this permission in their personal licence (held on their smart card). After the access profile has been transferred (preferably encrypted), preferably integrity checks are carried out and a digest of the profile is stored within the local TC.

If the data is sent to the smart card, it will preferably only be installed after an integrity check, and after a digest of the data is stored on the smart card. This will necessitate data protector-type code to be present on the smart card.

If the data is sent to the TC, it will preferably only be installed after an integrity check, and after a digest of the data is stored on the TC.

When the user logs into the PC using the smart card, the SC and TC mutually authenticate, and the TC stores the SC ID.

When the user wishes to access the data, perhaps via another program, the secure operator needs to check information stored locally such as an access profile associated with the data, or any licence-related information stored locally in order to see whether the user ID obtained during the last sign-on allows permission to carry out the required permission, or else whether there is generic permission to do so (irrespective or identity). (Preferably, the data protector will also check the integrity of the profile, and of the data.) If such permission is found, permission will be given to the operating system to access the data (optionally a check for a licence on the smart card will be made first). If not, the secure operator will query the smart card to find out if a licence is stored on the smart card relating to the data in question. If not, permission will be denied to the operating system to carry out the operation; if so, the licensing information will be retrieved by being encrypted via a shared session key and integrity checked (and possibly stored), and a check made to see whether the current operation is valid. If so, permission will be given to the operating system to access the data; if not, permission will be denied. Preferably, it will also check that the smart card corresponding to that user ID is still inserted in the smart card reader.

This approach could also be used where a smart card is duplicated or shared amongst members of a group.

There is the possibility to strengthen this approach by storing part of the data, or its functionality, within the TC itself so that what is stored on disk is incomplete, as described in a previous patent [licensing, API calls]. The most important part of the package is probably the wrapper (cf. software profile), and particular attention should be paid to protecting this. The wrapper associated with data could even be stored in clear on the TC or smart card at the time when the data was installed, and the data protector not allow the associated data to be executed unless an associated wrapper were located within the TC or smart card.

Whenever data is to be installed onto the trusted platform, integrity and other checks should be carried out in order to safely download or upgrade data from a third party. Data installation will only proceed via the operating system ('OS') if such the expected integrity values match. If such checks succeed (in the sense of the data or wrapper not having been altered), the data protector will store in the TC (e.g. smart card) the digest of the data (and any access profile) which was appended to the data itself, together with a reference to the stored data. Optionally, an alternative data form used in integrity checking such as the integrity checksum of the data is stored instead in the TC (e.g. smart card).

Any subsequent call for execution or other use of the data will always be passed to the TC. Preferably, before a check on the access profile is carried out by the secure operator, the data protector will make a check and will not allow the data to be executed if a hash, checksum or other digest of the data to be executed (calculated within the TC using its cryptographic capabilities) does not exactly match the expected value stored within the TC. This process is illustrated in Figure 5.

If the access profile and data is altered, it may not be possible to match the data against the digest stored within the TC, as it might not be clear what is the corresponding entry. Hence, again, preferably the data protector will not allow any data to be executed if there is not a corresponding correct digest stored within it.

Optionally, it is not permitted for data without a wrapper to be executed, to give further protection against the data being executed if the wrapper is removed.

Optionally, checks are also made by the data protector to ensure that multiple copies of the data are not in existence; this prevents for example unauthorised extension of usage of data protected by licensing models involving a set number of uses, or executing for a given time. If multiple copies are found, the user would be given the option to delete all copies except one, in order to allow execution of this copy.

Next the way in which these components interact to form a system for restricting usage of data will be considered. There are several stages in which such a system can be constructed, which may be considered as progressing from one another. The first stage is to use generic operation protection software (as detailed above) that performs checks upon operations applied to data and checks against unauthorised alteration and is protected against bypassing by integrity checking. Such operation protection software need not run within the trusted module itself. A preferred stage is the logical extension of such a system in which the operation protection software runs within the trusted module. A request to perform an operation upon some data will be sent to the trusted module, preferably from the access profile. The operation protection software in the trusted module will evaluate such a request and decide whether to allow this, based on the restrictions defined within the access profile. Preferably, the trusted module and an operating system of the platform have a dedicated communications path between them which is inaccessible to other parts of the computer platform. In the preferred model, the request from the secure operator to the operating system to access the data is preferably supplied via the dedicated communications path.

Preferably the trusted module is operable to log the request to the operating system to use the data. The security and reliability of metering of data usage is enhanced by securely logging data usage within the trusted module. Logging of data manipulation activity is carried out and recorded securely in the TC. There is the option to carry this out at a number of different stages. The most common would be at the stage at which the data was opened, copied, printed or allowed to run by the secure operator. Another common point would be at the stage at which the data protector has successfully completed its integrity checks on the data to be installed, and has successfully installed this data onto the client machine. Since the access profile, secure operator and data protector are protected by integrity checks, some protection is given against hackers trying to bypass or edit the logging process. Such logs would provide both secure auditing information and the possibility of flexible licensing and payment models. Such audit logs would form the basis for usage reports and information accessible to third parties such as the machine user's IT department or company auditors.

In order to counter software piracy, by giving protection against use of copied versions of the data outside the trusted platform, there are several approaches, corresponding to techniques used within dongle technology today. First, the data itself can be transmitted and stored encrypted, with the decryption key stored in the access profile or within the licence stored on the smart card. Secondly, API calls could be inserted into the data, if the source code were available, to check for the TC ID or the smart card ID, or a key stored within the TC or SC, before data access permission is given and/or during the data

access operation. Such measures are not necessary if the only aim is to ensure that the data cannot be accessed on trusted platforms in a manner outside the licence agreements with the developer.

A preferred embodiment of the present invetion will now be described with reference to the accompanying drawings.

Figure 1 illustrates the physical system. The Trusted Component [103] is in the path between normal computer host [104] and the display [101]. This enables the TC [103] to reliably write to the display, without fear of subversion from normal software, including the operating system. The host computer is connected to a keyboard [105] that has a built-in smart card reader. A smart card [106] is plugged into the keyboard and can be accessed by the host system [104] and the TC [103]. The smart card [106] is able to communicate securely with the TC [103]. (It should be noted that this architecture is that described in EP 99301100.6 and is not an essential part of this invention.)

Figure 2 illustrates a logical diagram of the TC [103] components, comprising operation protection software components [211] and other operation protection data components [210] within the TC [103]. The following components of the invention are operation protection code [211] that should be run within a protected environment, as previously described, and preferably within the TC itself [103]: the secure operator [206], and the data protector [207]. Operation protection data components stored on the TC include the private key of the TC [201], the public key certificate of a trusted entity [202], the developer's public key certificate [203], a log [204], and a hashed version of [206]-[207], signed by the trusted entity [205].

Figure 3 illustrates the structure of protected software or data [306] within the client computer [104]. Digital data [304] on the client computer [102] is associated with a access profile [303], within which is stored the public key of the TC [302]. This structure [301] is stored together with a hashed version of [301], signed with the clearinghouse or developer's private key [305]. Preferably, the hashed version is stored within the TC itself (this is carried out during the installation process by the data protector [207]).

Figure 4 illustrates the flowchart for loading or upgrading software or other data onto the client platform, for the general case where the data protector may not be running within the TC.

The data to be installed is hashed and signed with the sender's private key, and this is appended to the data itself by the sender.

When the operating system [400] requests that the data should be installed, the protector [207] checks the signature of this message, using the public key certificate corresponding to the sender, thereby checking authentication of the sender.

If authentication fails, the data protector sends an error message to the operating system and the operating system [400] displays an appropriate message.

If authentication succeeds, the data protector computes the hash of the message, via the cryptographic capabilities available within the TC [103] and compares it to the message hash that is associated with the data. This checks for integrity of the message.

If the hashes are the same, the data protector saves a hash of the message within the TC and indicates that the OS can install the data as normal, and the TC makes a log of the installation.

If the hashes are not the same, this indicates that the data has been altered, and that it should not be installed. The data protector sends an error message to the OS, which displays an appropriate message to the user.

Figure 5 illustrates the relationship between each portable trusted module (henceforth just referred to as SC, using the particular example of a smart card [106]) and TC [103]. There is mutual authentication at sign-on and the TC checks the SC's ID (preferably via a certificate of the SC's public key). Before the secure operator on the TC refuses permission to the OS to access data, the TC makes a check for a relevant user licence on the smart card. The OS is modified in the sense that it has trusted input/output process for data, which can be strengthened by a secure hardware communications path between the secure operator software and the OS that is not accessible to other software. This part of the OS will be checked upon BIS, and there is an option for the system integrity check to fail if the integrity check on this part of the OS fails.

Figure 6 illustrates the flowchart for operation restriction using a model of checking where the OS communicates with the secure operator which is (in) TC, and the access profile outside TC associated with a piece of data specifies the operations allowed by the developer upon that data. This is for the general case where preferably, all operation protection software is mounted within the TC. Communications between the OS, the operation protection software and the TC need to be protected against modification or spoofing. One option is to make part of the OS trusted (the part dealing with data access i.e. data input and output), and this part of the OS can be integrity checked as part of the BIS procedure. If this part has been modified, then the platform integrity will fail. Another option is to use a communication path (hardware, and preferably dedicated) from the TC to the CPU when communicating with the OS. The procedure is as follows:

Upon sign-on using the smart card, there is mutual authentication between the TC and the smart card. The TC stores the (current) smart card ID, which is preferably the certificate of the smart card public key.

When the user wishes to carry out an operation on some digital data, preferably the OS [400] sends a message to the data protector [207], which checks whether there is a hash or checksum corresponding to the data or to the issuer of the data and access profile stored within the TC.

If there is not, the data protector relays a message to the operating system and the data is not executed.

If there is, the secure operator [206] issues a challenge/response to the access profile [303] corresponding to that piece of data, by means of sending a random number (nonce), together with a reference to the data (e.g. its title), signed using the private key of the TC.

The access profile verifies and authenticates the secure operator's challenge using the public key of the TC [103], and returns a message incorporating the nonce and reference to the data. The nonce is included to give protection against replay attacks.

The secure operator will make the appropriate operating check dependent upon the information contained within the access profile, with reference to the smart card ID and the TC ID.

If the access profile signals an error because it does not wish the data operation to be carried out on this particular machine by any user, the secure operator relays a message to the operating system and the data is not operated upon.

If there is no access profile associated with the data, the secure operator makes a decision about whether to allow the operation corresponding to stage (j) below or to a default model previously set within it by an administrator. For example, the administrator may wish to stipulate that deletion can occur by default, but that copying more than a certain number of times cannot.

If no explicit access permissions are given, and optionally the access profile contains a flag that licences can be checked for this type of data access, the secure operator issues a challenge/response to the smart card, by means of sending a nonce, together with a reference to the data, signed using the private key of the TC. The smart card then verifies and authenticates the secure operator's challenge using the public key of the TC, and returns a message incorporating the nonce, reference to data and user access licence information. The secure operator then checks for appropriate permission within this licence to carry out the data access operation.

If there is no valid permission within the access profile, the secure operator asks the OS [400] to notify the end-user appropriately and the data is not operated upon.

If there is a valid permission within the access profile, the secure operator asks OS to carry out the data operation and the TC [103] takes a metering record of the transaction.

Optionally, the access profile can make the initial request to access data (optionally specifying profile checks) to the TC, which can then carry out integrity checks on the data and profile, and check for licence permissions.

There is a variation on this procedure in which the profile is more proactive, and the OS [400] contacts the access profile directly to request an operation on data, and the profile responds with permission only in the case where the operation does not counter the profile specification. Similarly, the user licence on the smart card can initiate the appropriate checks.

A general procedure for usage checking that uses a proactive profile within the smart card, together with checking carried out by TC on the PC, would be as follows:

Upon registration and/or payment for the data, the clearinghouse or developer (according to the exact payment model) authorises the licence corresponding to the smart card ID and data to be updated, according to the data purchased. (Prior to this, there will be mutual authentication (possibly off-line), and public key certificates between these bodies will have been exchanged, or else the developer will actually issue the smart card). The clearinghouse or developer sends the data, associated with a (customised) access profile, to the client. The access profile is customised such that the public key of the smart card is inserted into the access profile (alternatively, a shared key is set up between the secure operator and the smart card). Both the data and the access profile are hashed and signed with the clearinghouse/developer's private key, and the public key corresponding to this is stored on the smart card. The contents of any message which is to be protected are encrypted using a randomly generated secret key (such as a DES key), and transferred together with the symmetric key which is public key (e.g. RSA)-encrypted using the public key of the intended recipient, according to a standard protocol. If the data is transferred to the PC, an analogous process is carried out for transferral of the data, with the public key of the developer being sent to the TC.

The data protector integrity checks the data whenever this is transferred to the TC: upon installation, the package is verified by hashing and comparison with the decrypted signature (using the public key in the TC), and a hash is stored in the TC. The data is not loaded if the digital signature does not match what is expected.

The access profile is not loaded if the digital signature does not match what is expected.

Upon sign-on using the smart card, there is mutual authentication between the TC and the smart card.

The TC stores the (current) smart card ID.

When the user wishes to use some data, the access profile corresponding to that data issues a challenge/response to the secure operator, by means of sending a random number (nonce), together with a reference to the data.

The secure operator makes an appropriate check on the data, using the smart card ID, or else by obtaining some information stored on the smart card. For example, the secure operator:

checks whether the data is licensed to be used according to the user ID of the smart card which has been inserted, in the profile stored within the TC, or

checks whether the data is licensed to be used on the TC ID according to the profile stored within the TC, or

consults the smart card to obtain further details of any licence stored therein associated with the data to be accessed. In this case, the secure operator issues a challenge/response to the smart card, by means of sending a nonce, together with a reference to the data, signed using the private key of the TC. The smart card then verifies and authenticates the secure operator's challenge using the public key of the TC, and returns a message incorporating the nonce, reference to data and user access licence information. The secure operator then checks for appropriate permission within this licence to carry out the data access operation.

If there is no valid licence, the secure operator returns an error message, from which the access profile can determine the exact type of problem with licensing and notify the OS appropriately. If there is a valid licence, the secure operator returns a message incorporating the nonce and data reference, signed and encrypted using the TC's private key.

The access profile verifies if the secure operator's reply is correct using the TC's public key, and either passes the call to the OS to execute the data or sends an error message to the OS as appropriate.

The log is preferably held within the machine TC, but could in addition or instead be held in the smart card, and is updated appropriately.

There are analogous models for such an access control process corresponding to having a less proactive access profile, as described in Figure 5.

Figure 6 illustrates the flowchart for data protection, where operation protection software is stored within the TC and the TC acts as a bridge between an application and the OS. The process is similar to that given in Figure 5, except that the secure operator and

data protector are within the TC itself and the secure operator uses a communication (preferably dedicated) from the TC to the CPU when communicating with the OS.

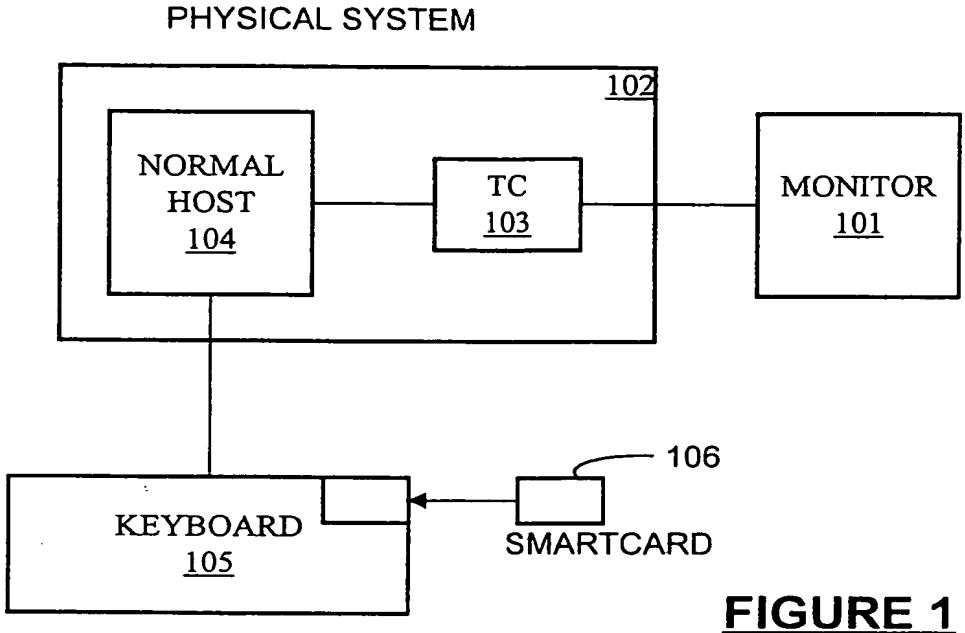
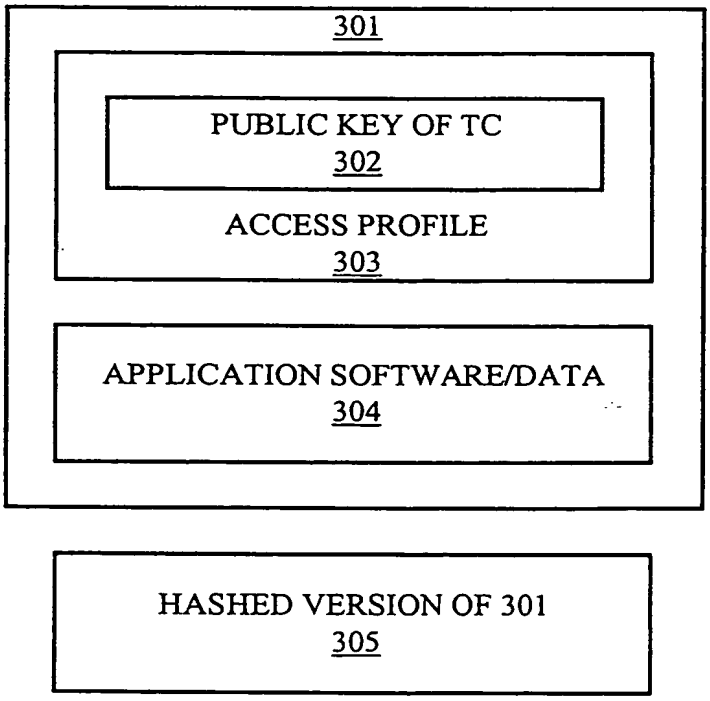
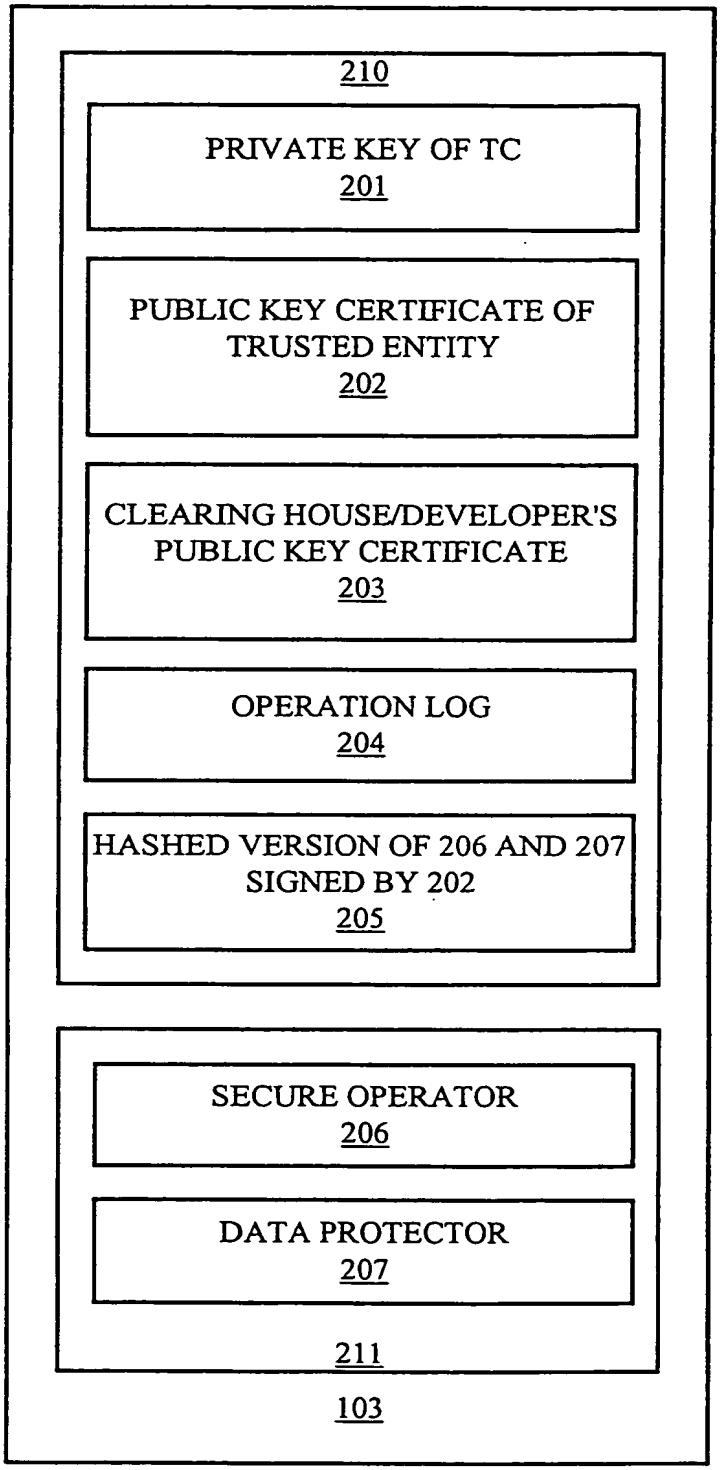


FIGURE 1



LOGICAL DIAGRAM OF APPLICATION SOFTWARE/DATA MOUNTED ON CLIENT PC

FIGURE 3



LOGICAL DIAGRAM OF TC

FIGURE 2

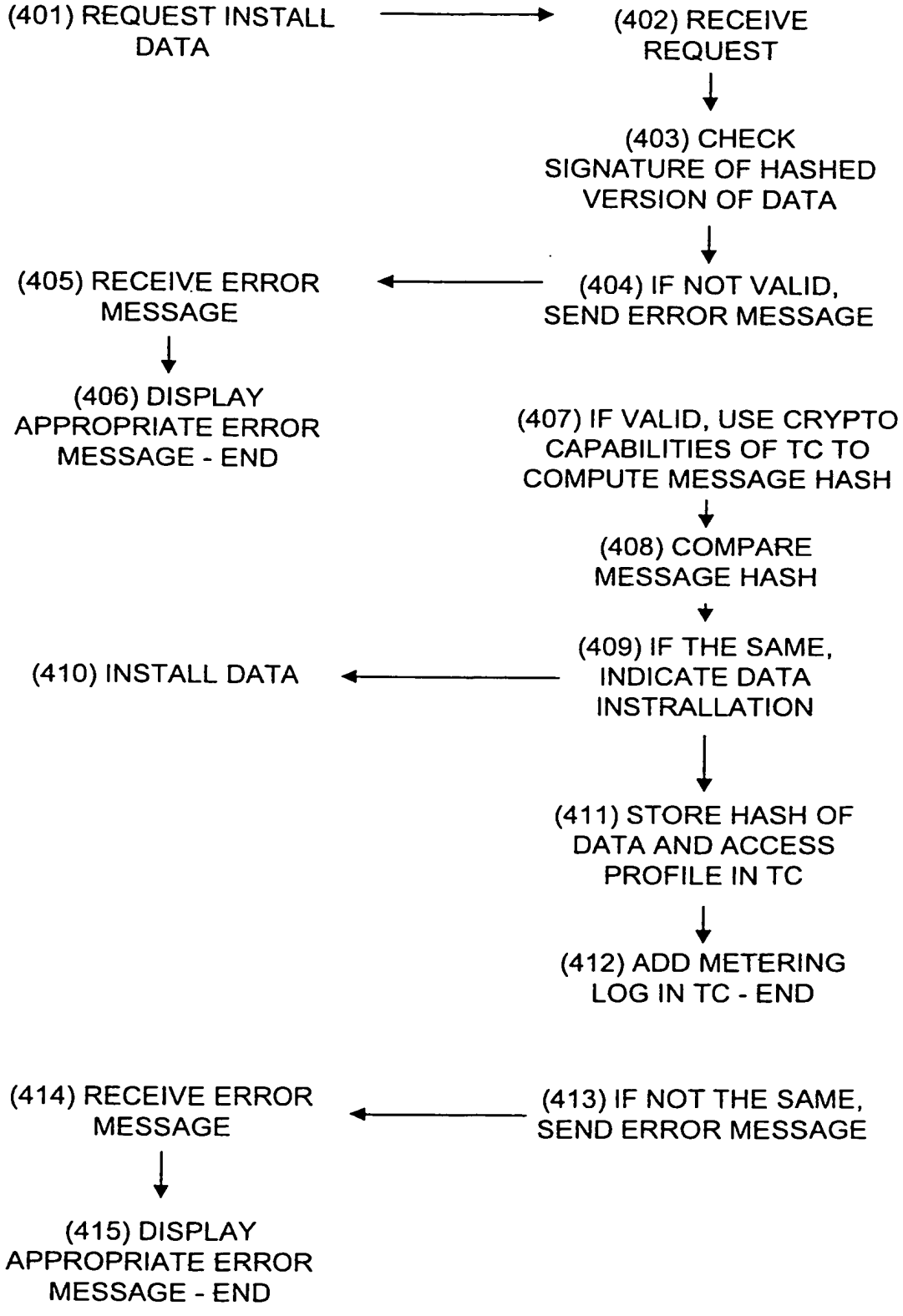
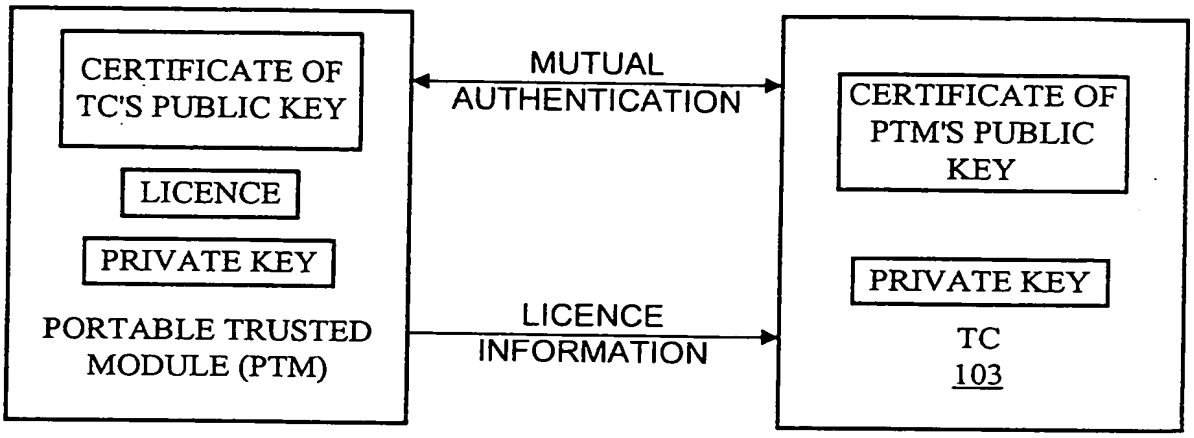

OS 400
DATA PROTECTOR 207
(401) REQUEST INSTALL DATA
(402) RECEIVE REQUEST
(403) CHECK SIGNATURE OF HASHED VERSION OF DATA
(404) IF NOT VALID, SEND ERROR MESSAGE
(405) RECEIVE ERROR MESSAGE
(406) DISPLAY APPROPRIATE ERROR MESSAGE - END
(407) IF VALID, USE CRYPTO CAPABILITIES OF TC TO COMPUTE MESSAGE HASH
(408) COMPARE MESSAGE HASH
(409) IF THE SAME, INDICATE DATA INSTRALLATION
(410) INSTALL DATA
(411) STORE HASH OF DATA AND ACCESS PROFILE IN TC
(412) ADD METERING LOG IN TC - END
(413) IF NOT THE SAME, SEND ERROR MESSAGE
(414) RECEIVE ERROR MESSAGE
(415) DISPLAY APPROPRIATE ERROR MESSAGE - END

FIGURE 4



RELATIONSHIP BETWEEN A PORTABLE TRUSTED MODULE AND A PC'S TC

FIGURE 5

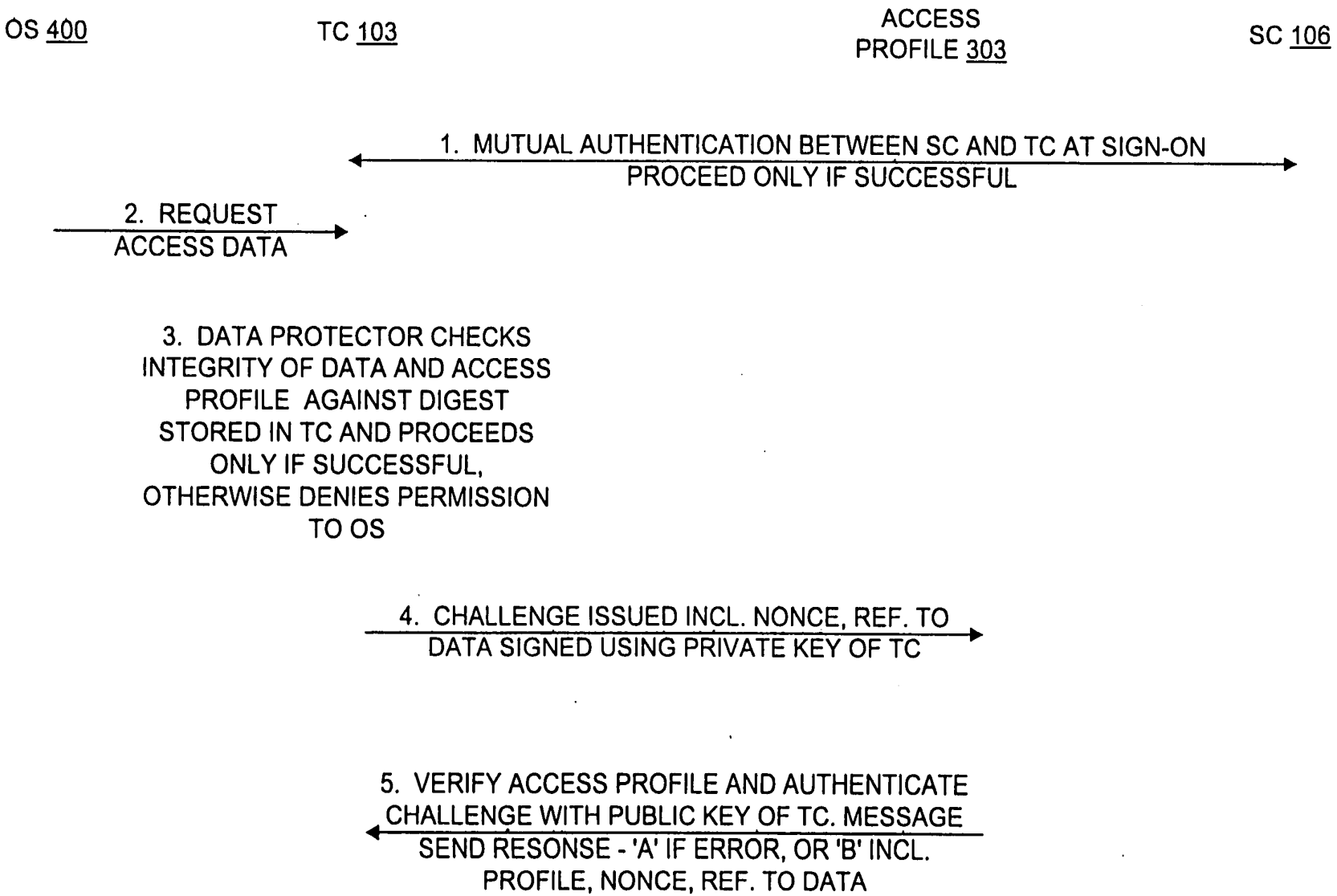

OS 400
TC 103
ACCESS PROFILE 303
SC 106
1. MUTUAL AUTHENTICATION BETWEEN SC AND TC AT SIGN-ON
PROCEED ONLY IF SUCCESSFUL
2. REQUEST ACCESS DATA
3. DATA PROTECTOR CHECKS INTEGRITY OF DATA AND ACCESS PROFILE AGAINST DIGEST STORED IN TC AND PROCEEDS ONLY IF SUCCESSFUL, OTHERWISE DENIES PERMISSION TO OS
4. CHALLENGE ISSUED INCL. NONCE, REF. TO DATA SIGNED USING PRIVATE KEY OF TC
5. VERIFY ACCESS PROFILE AND AUTHENTICATE CHALLENGE WITH PUBLIC KEY OF TC. MESSAGE SEND RESONSE - 'A' IF ERROR, OR 'B' INCL. PROFILE, NONCE, REF. TO DATA

FIGURE 6

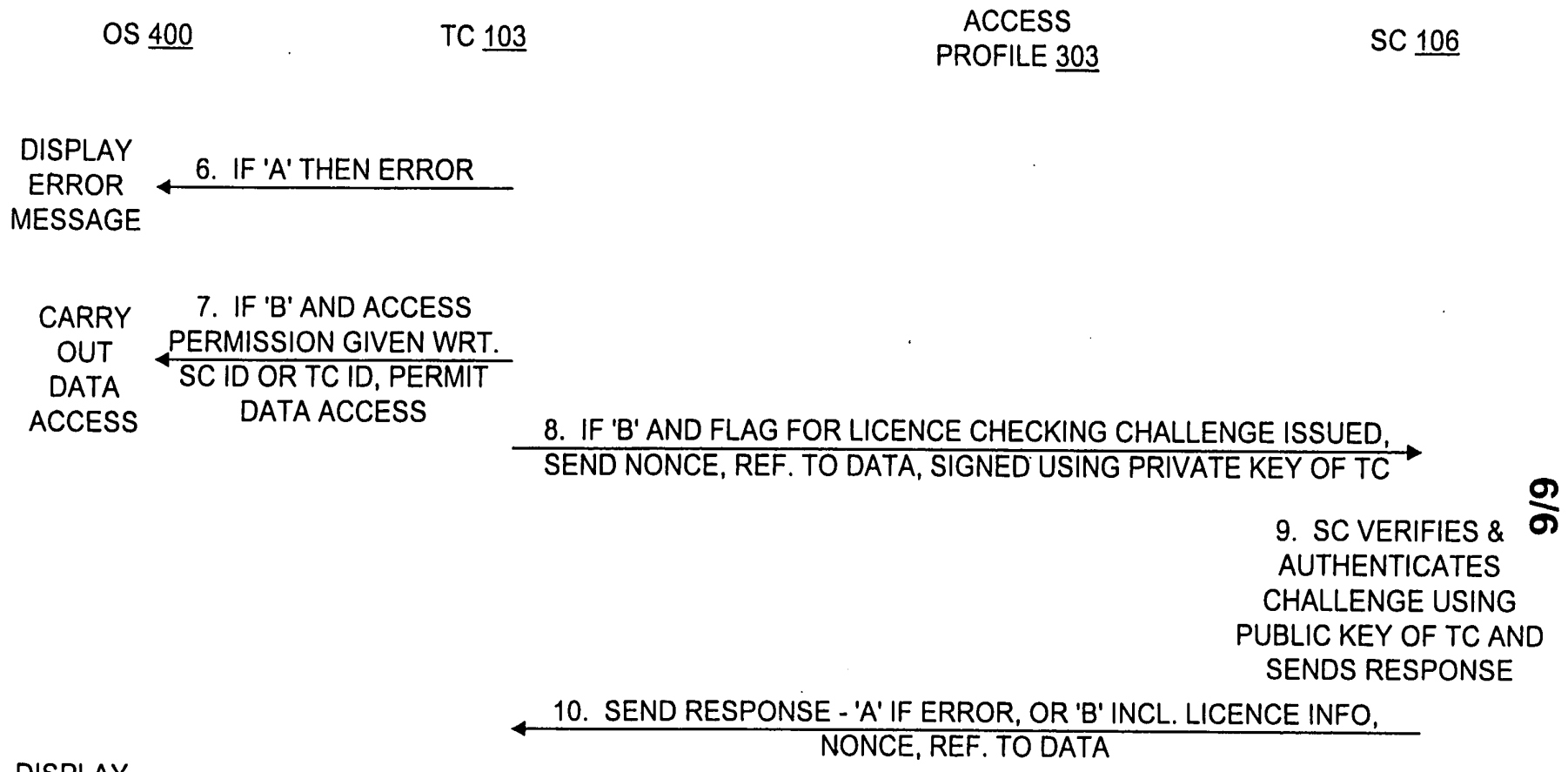


FIGURE 6 (CONT.)

Trusted Computing Platform

(HP Ref 30990050)

Technical Field

The present invention generally relates to trusted devices, trusted computing platforms, trusted transactions and methods of operating the same.

Background Art

For commercial applications, a client computing platform typically operates in an environment where its behaviour is vulnerable to modification by local or remote entities. This potential insecurity of the platform is a limitation on its use by local parties who might otherwise be willing to use the platform, or remote parties who might otherwise communicate with the platform; for example, for the purposes of E-commerce. For the present purposes, both local parties and remote parties will be referred to as "users" unless otherwise stated.

Existing security applications, for example virus detection software, execute on computing platforms under the assumption that the platform will operate as intended and that the platform will not subvert processes and applications. This is a valid assumption provided that the intended software state has not become unstable or has not been damaged by other software such as viruses. Users, therefore, typically restrict the use of such platforms to non-critical applications, and weigh the convenience of the using the platforms against the risk to sensitive or business critical data.

Increasing the level of trust in platforms therefore enables greater user confidence in existing security applications (such as the 'Secure Sockets Layer' or 'IPSec') or remote management applications. This enables greater reliance on those applications and hence reduced 'cost of ownership'. Greater trust also enables new electronic methods of business, since there is greater confidence in the correct operation of both local and remote computing platforms.

In this document, the word 'trust' is used in the sense that something can be 'trusted' if it always behaves in the way it is expected to behave.

Disclosure of the Invention

The present inventors have appreciated that it is desirable to use a physical device in a computing platform to verify and possibly enforce trust in that platform. Typically, the device provides trusted measurement and reporting of attributes of the associated platform, which indicate the integrity of the platform. Also, most preferably, the device is tamper-resistant.

In accordance with a first aspect, the present invention provides computing apparatus comprising mounted on an assembly main processing means and main memory means, each being connected for communication with one or more other components on the assembly,

characterised by further comprising a trusted device mounted on the assembly and being connected for communications with one or more other components on the assembly, the trusted device being arranged to acquire a true value of an integrity metric of the computing apparatus.

As used herein for reasons of simplicity of description, the term "device" also encompasses plural devices having equivalent function, or equivalent functionality integrated into one or more existing platform devices or assemblies. Additionally, the term 'true' as used herein implies that the value is that which correctly reflects the state of the computing apparatus. This may be ensured if the measurement method is substantially un-modifiable other than by the trusted device.

In accordance with a second aspect, the present invention provides a method of operating a system comprising trusted computing apparatus and a user, the trusted computing apparatus incorporating a trusted device being arranged to acquire the true value of an integrity metric of the computing apparatus, the method comprising the steps of:

the trusted device acquiring the true value of the integrity metric of the trusted computing apparatus;

the user generating a challenge for the trusted computing apparatus to prove its integrity and submitting the challenge to the trusted computing apparatus;

the trusted computing apparatus receiving the challenge, and the trusted device generating a response including the integrity metric and returning the response to the user; and

the user receiving the response, extracting the integrity metric from the response and comparing the integrity metric with an authenticated metric for the trusted computing apparatus that had been generated by a trusted party.

In accordance with a third aspect, the present invention provides a method of establishing a communications channel in a system between trusted computing apparatus and remote computing apparatus, the method including the step of the remote computing apparatus verifying the integrity of the trusted computing apparatus using the above method, and maintaining the communications channel for further transactions in the event the integrity of the trusted computing apparatus is successfully verified by the remote computing apparatus.

In accordance with a fourth embodiment, the present invention provides a method of verifying that trusted computing apparatus is trustworthy for use by a user for processing a particular application, the method including the step of the user verifying the integrity of the trusted computing apparatus using the above method, and the user using the trusted computing apparatus to process the particular application in the event the integrity of the trusted computing apparatus is successfully verified by the remote computing apparatus.

Other aspects and embodiments of the present invention will become apparent from the following description and claims.

Brief Description of the Drawings

A preferred embodiment of the present invention will now be described by way of example only with reference to the accompanying drawings in which:

Figure 1 is a diagram which shows the motherboard of computing apparatus adapted to include a trusted device according to an embodiment of the present invention;

Figure 2 is a diagram which shows in more detail the trusted device shown in Figure 1;

Figure 3 is a diagram which shows in the contents of a certificate stored in the trusted device;

Figure 4 is a diagram, which shows the features of a measurement function responsible for acquiring an integrity metric;

Figure 5 is a flow diagram which illustrates the steps involved in acquiring an integrity metric of the computing apparatus; and

Figure 6 is a flow diagram which illustrates the steps involved in establishing communications between a trusted computing platform and a remote platform including the trusted platform verifying its integrity.

Best Mode For Carrying Out the Invention, & Industrial Applicability

The present exemplary embodiment generally provides the incorporation into a computing platform of a physical trusted device whose function is to bind the identity of the platform to reliably measured data that provides an integrity metric of the platform. The identity and the integrity metric are compared with expected values provided by a trusted party (TP) that is prepared to vouch for the trustworthiness of the platform. If there is a match, the implication is that at least part of the platform is operating correctly, depending on the scope of the integrity metric.

A user verifies the correct operation of the platform before exchanging other data with the platform. A user does this by requesting the trusted device to provide its identity and an integrity metric. (Optionally the trusted device will refuse to provide evidence of identity if it itself was unable to verify correct operation of the platform.) The user receives the proof of identity and the identity metric, and compares them against values which it believes to be true. Those proper values are provided by the TP or another entity that is trusted by the user. If data reported by the trusted device is the same as that provided by the TP, the user trusts the platform. This is because the user trusts the entity. The entity trusts the platform because it has previously validated the identity and determined the proper integrity metric of the platform.

Once a user has established trusted operation of the platform, he exchanges other data with the platform. For a local user, the exchange might be by interacting with some software application running on the platform. For a remote user, the exchange might involve a secure transaction. In either case, the data exchanged is 'signed' by the trusted device. The user can then have greater confidence that data is being exchanged with a platform whose behaviour can be trusted.

The trusted device uses cryptographic processes but does not necessarily provide an external interface to those cryptographic processes. Also, a most desirable implementation would be to make the trusted device tamperproof, to protect secrets by making them inaccessible to other platform functions and provide an environment that is substantially immune to unauthorised modification. Since tamper-proofing is impossible, the best approximation is a trusted device that is tamper-resistant, or tamper-detecting. The trusted device, therefore, preferably consists of one physical component that is tamper-resistant.

Techniques relevant to tamper-resistance are well known to those skilled in the art of security. These techniques include methods for resisting tampering, methods for detecting tampering, and methods for eliminating data when tampering is detected. It will be appreciated that, although tamper-proofing is a most desirable feature of the present invention, it does not enter into the normal operation of the invention and, as such, is beyond the scope of the present invention and will not be described in any detail herein.

The trusted device is preferably a physical one because it must be difficult to forge. It is most preferably tamper-resistant because it must be hard to counterfeit. It typically has an engine capable of using cryptographic processes because it is required to prove identity, both locally and at a distance, and it contains at least one method of measuring some integrity metric of the platform with which it is associated.

Figure 1 illustrates the motherboard 10 of an exemplary computer platform (not shown). The motherboard 10 includes (among other standard components) a main processor 11, main memory 12, a trusted device 14, a data bus 16 and respective standard control lines 17 and address lines 18, and BIOS memory 19 containing the BIOS program for the platform.

Typically, the BIOS program is located in a special reserved memory area, the upper 64K of the first megabyte do the system memory (addresses FØØØh to FFFFh), and the main processor is arranged to look at this memory location first, in accordance with an industry wide standard

The significant difference between the platform and a conventional platform is that, after reset, the main processor is initially controlled by the trusted device, which then hands control over to the platform-specific BIOS program, which in turn initialises all input/output devices as normal. After the BIOS program has executed, control is handed over as normal by the BIOS program to an operating system program, such as Windows NT (TM), which is typically loaded into main memory 12 from a hard disk drive (not shown).

Clearly, this change from the normal procedure requires a modification to the implementation of the industry standard, whereby the main processor 11 is directed to address the trusted device 14 to receive its first instructions. This change may be made simply by hard-coding a different address into the main processor 11. Alternatively, the trusted device 14 may be assigned the standard BIOS program address, in which case there is no need to modify the main processor configuration.

Although, in the preferred embodiment to be described, the trusted device 14 is a single, discrete component, it is envisaged that the functions of the trusted device 14 may alternatively be split into multiple devices on the motherboard, or even integrated into one or more of the existing standard devices of the platform. For example, it is feasible to integrate one or more of the functions of the trusted device into the main processor itself, provided that the functions and their communications cannot be subverted. This, however, would probably require separate leads on the processor for sole use by the trusted functions. Additionally or alternatively, although in the present embodiment the trusted device is a hardware device that is adapted for integration into the motherboard 10, it is anticipated that a trusted device may be implemented as a 'removable' device, such as a dongle, which could be attached to a platform when required. Whether the trusted device is integrated or removable is a matter of design choice.

The trusted device 14 comprises a number of blocks, as illustrated in Figure 2: a controller 20 for controlling the overall operation of the trusted device 14, and interacting with

the other functions on the trusted device 14 and with the other devices on the motherboard 10; a measurement function 21 for acquiring an integrity metric from the platform; a cryptographic function 22 for signing or encrypting specified data; and interface circuitry 23 having appropriate ports (24, 25 & 26) for connecting the trusted device 14 respectively to the data bus 16, control lines 17 and address lines 18 of the motherboard 10. Each of the blocks in the trusted device 14 has access (typically via the controller 20) to appropriate volatile memory areas 27 and/or non-volatile memory areas 28 of the trusted device 14.

For reasons of performance, the trusted device 14 may be implemented as an application specific integrated circuit (ASIC). However, for flexibility, the trusted device is preferably an appropriately programmed micro-controller. Both ASICs and micro-controllers are well known in the art of microelectronics and will not be considered herein in any further detail.

One item of data stored in the non-volatile memory is a certificate 30, which is illustrated in Figure 3. The certificate 30 contains at least a public key 32 of the trusted device 14 and an authenticated value of a platform integrity metric 34 measured by a TP. Optionally, the trusted device 14 also contains an identity (ID) label 36 of the trusted device 14.

Where present, the ID label 36 is a conventional ID label, for example a serial number, that is unique within some context. The ID label 36 is generally used for indexing and labelling of data relevant to the trusted device 14, but is insufficient in itself to prove the identity of the platform under trusted conditions.

The trusted device 14 is equipped with at least one method of reliably measuring some integrity metric of the computing platform with which it is associated. The integrity metric is acquired by the measurement function 21, which is illustrated in more detail in Figure 4.

The measurement function 21 has access to non-volatile memory 40 for storing a hash program 41, plus volatile memory 42 for storing a computed integrity metric 43, in the form of a digest. The hash program 41 contains instructions for computing the digest, in code that is native to the main processor 11. In addition, part of the measurement function 21 is configured to respond to the main processor 11 as if it were addressable memory, such as standard read-only memory, by sensing memory read signals addressed to the trusted device 14 and returning appropriate data. The result is that the main processor 11 sees the trusted device, for the purposes of integrity metric measurement, as a standard read-only memory.

In the preferred implementation, as well as the digest, the integrity metric includes a Boolean value 44, which is stored in volatile memory 45 by the measurement function 21, for reasons that will become apparent.

A preferred process for acquiring an integrity metric will now be described with reference to Figure 5.

In step 500, at switch-on, the measurement function 21 monitors the activity of the main processor 11 on the data, control and address lines (16, 17 & 18) to determine whether the trusted device 14 is the first memory accessed. Under conventional operation, a main processor would first be directed to the BIOS memory first in order to execute the BIOS program. However, in accordance with the present embodiment, the main processor 11 is directed to the trusted device 14, which acts as a memory. In step 505, if the trusted device 14 is the first memory accessed, in step 510, the measurement function 21 writes to non-volatile memory 45 a Boolean value 44, which indicates that the trusted device 14 was the first memory accessed. Otherwise, in step 515, the measurement function writes a Boolean value 44, which indicates that the trusted device 14 was not the first memory accessed.

In the event the trusted device 14 is not the first accessed, there is of course a chance that the trusted device 14 will not be accessed at all. This would be the case, for example, if the main processor 11 were manipulated to run the BIOS program first. Under these circumstances, the platform would operate, but would be unable to verify its integrity on demand, since the integrity metric would not be available. Further, if the trusted device 14 were accessed after the BIOS program had been accessed, the Boolean value 44 would clearly indicate lack of integrity of the platform.

In step 520, when (or if) accessed as a memory by the main processor 11, the main processor 11 reads the stored native hash instructions 41 from the measurement function 21 in step 525. The hash instructions 41 are passed for processing by the main processor 11 over the data bus 16. In step 530, main processor 11 executes the hash instructions 41 and uses them, in step 535, to compute a digest of the BIOS memory 19, by reading the contents of the BIOS memory 19 and processing those contents according to the hash program. In step 540, the main processor 11 writes the computed digest 43 to the appropriate non-volatile memory location 42 in the trusted device 14. The measurement function 21, in step 545, then calls the BIOS program in the BIOS memory 19, and execution continues in a conventional manner.

Clearly, there are a number of different ways in which the integrity metric may be calculated, depending upon the scope of the trust required. The measurement of the BIOS program's integrity provides a fundamental check on the integrity of a platform's underlying

processing environment. Other integrity checks could involve establishing that various other devices, components or apparatus attached to the platform are present and in correct working order. In one example, the BIOS programs associated with a SCSI controller could be verified to ensure communications with peripheral equipment could be trusted. In another example, the integrity of other devices, for example memory devices or co-processors, on the platform could be verified by enacting fixed challenge/response interactions to ensure consistent results. Also, although in the present embodiment the trusted device 14 utilises the data bus as its main means of communication with other parts of the platform, it would be feasible, although not so convenient, to provide alternative communications paths, such as hard-wired paths or optical paths. Further, although in the present embodiment the trusted device 14 instructs the main processor 11 to calculate the integrity metric, it is anticipated that, in other embodiments, the trusted device itself will be arranged to measure one or more integrity metrics.

Preferably, the BIOS boot process includes mechanisms to verify the integrity of the boot process itself. Such mechanisms are already known from, for example, Intel's draft "Wired for Management baseline specification v 2.0 - BOOT Integrity Service", and involve calculating digests of software or firmware before loading that software or firmware. Such a computed digest is compared with a value stored in a certificate provided by a trusted entity, whose public key is known to the BIOS. The software/firmware is then loaded only if the computed value matches the expected value from the certificate, and the certificate has been proven valid by use of the trusted entity's public key. Otherwise, an appropriate exception handling routine is invoked.

Optionally, after receiving the computed BIOS digest, the trusted device 14 may inspect the proper value of the BIOS digest in the certificate and not pass control to the BIOS if the computed digest does not match the proper value. Additionally, or alternatively, the trusted device 14 may inspect the Boolean value 44 and not pass control back to the BIOS if the trusted device 14 was not the first memory accessed.

Figure 6 illustrates the flow of actions by a TP, the trusted device 14 incorporated into a platform, and a user (of a remote platform) who wants to verify the integrity of the trusted platform. It will be appreciated that substantially the same steps as are depicted in Figure 6 are involved when the user is a local user. In either case, the user would typically rely on some form of software application to enact the verification. It would be possible to run the software application on the remote platform or the trusted platform. However, there is a chance that, even on the remote platform, the software application could be subverted in some way. Therefore, it is anticipated that, for a high level of integrity, the software

application would reside on a smart card of the user, who would insert the smart card into an appropriate reader for the purposes of verification.

At the first instance, a TP, which vouches for trusted platforms, will inspect the type of the platform to decide whether to vouch for it or not. This will be a matter of policy. If all is well, in step 600, the TP measures the value of integrity metric of the platform. Then, the TP generates a certificate, in step 605, for the platform. The certificate is generated by the TP by appending the trusted device's public key, and optionally its ID label, to the measured integrity metric, and signing the string with the TP's private key.

The trusted device 14 can subsequently prove its identity by using its private key to process some input data received from the user and produce output data, such that the input/output pair is statistically impossible to produce without knowledge of the private key. Hence, knowledge of the private key forms the basis of identity in this case. Clearly, it would be feasible to use symmetric encryption to form the basis of identity. However, the disadvantage of using symmetric encryption is that the user would need to share his secret with the trusted device. Further, as a result of the need to share the secret with the user, while symmetric encryption would in principle be sufficient to prove identity to the user, it would insufficient to prove identity to a third party, who could not be entirely sure the verification originated from the trusted device or the user.

In step 610, the trusted device 14 is initialised by writing the certificate 30 into the appropriate non-volatile memory locations of the trusted device 14. This is done, preferably, by secure communication with the trusted device 14 after it is installed in the motherboard 10. The method of writing the certificate to the trusted device 14 is analogous to the method used to initialise smart cards by writing private keys thereto. The secure communications is supported by a 'master key', known only to the TP, that is written to the trusted device (or smart card) during manufacture, and used to enable the writing of data to the trusted device 14; writing of data to the trusted device 14 without knowledge of the master key is not possible.

At some later point during operation of the platform, for example when it is switched on or reset, in step 615, the trusted device 14 acquires and stores the integrity metric 42 of the platform.

When a user wishes to communicate with the platform, in step 620, he creates a nonce, such as a random number, and, in step 625, challenges the trusted device 14 (the operating system of the platform, or an appropriate software application, is arranged to recognise the challenge and pass it to the trusted device 14, typically via a BIOS-type call, in an appropriate fashion). The nonce is used to protect the user from deception caused by

replay of old but genuine signatures (called a 'replay attack') by untrustworthy platforms. The process of providing a nonce and verifying the response is an example of the well-known 'challenge/response' process.

In step 630, the trusted device 14 receives the challenge and creates a digest of the measured integrity metric and the nonce, and optionally its ID label. Then, in step 635, the trusted device 14 signs the digest, using its private key, and returns the signed digest, accompanied by the certificate 30, to the user.

In step 640, the user receives the challenge response and verifies the certificate using the well known public key of the TP. The user then, in step 650, extracts the trusted device's 14 public key from the certificate and uses it to decrypt the signed digest from the challenge response. Then, in step 660, the user verifies the nonce inside the challenge response. Next, in step 670, the user compares the computed integrity metric, which it extracts from the challenge response, with the proper platform integrity metric, which it extracts from the certificate. If any of the foregoing verification steps fails, in steps 645, 655, 665 or 675, the whole process ends in step 680 with no further communications taking place.

Assuming all is well, in steps 685 and 690, the user and the trusted platform use other protocols to set up secure communications for other data, where the data from the platform is preferably signed by the trusted device 14.

The techniques of signing, using certificates, and challenge/response, and using them to prove identity, are well known to those skilled in the art of security and will, thus, not be described in any more detail herein.

CLAIMS

1. Computing apparatus comprising mounted on an assembly main processing means and main memory means, each being connected for communication with one or more other components on the assembly,

characterised by further comprising a trusted device mounted on the assembly and being connected for communications with one or more other components on the assembly, the trusted device being arranged to acquire a true value of an integrity metric of the computing apparatus.

2. Computing apparatus according to claim 1, wherein the trusted device comprises device memory means and means for instructing the main processing means to determine the integrity metric and return the integrity metric for storage in the device memory means.

3. Computing apparatus according to claim 2, wherein the means for instructing the main processing means comprises, stored in the device memory means, program code native to the main processing means, and the trusted device is arranged to transfer the instructions of the program code to the main processing means.

4. Computing apparatus according to claim 3, wherein the platform is arranged to cause the instructions to be the first instructions executed after release from reset.

5. Computing apparatus according to claim 3 or claim 4, wherein the trusted device is arranged to transfer the instructions to the main processing means in response to memory read signals from the main processing means.

6. Computing apparatus according to any one of claims 1 to 5, wherein the trusted device comprises device memory means and is arranged to monitor the data bus means and store in the device memory means a flag in the event the first memory read signals generated by the main processing means after the computing apparatus is released from reset are addressed to the trusted device.

7. Computing apparatus according to any one of claims 1 to 6, wherein the trusted device has stored in device memory means at least one of:

a unique identity of the trusted device;

an authenticated integrity metric generated by a trusted party; and

a secret.

8. Computing apparatus according to claim 7, wherein the trusted device has stored in device memory means a secret comprising a private asymmetric encryption key.

9. Computing apparatus according to claim 8, wherein the trusted device also has stored in device memory means a respective public encryption key that has been signed by a trusted party.

10. Computing apparatus according to claim 8 or claim 9, wherein the trusted device has stored in device memory means an authenticated integrity metric generated by a trusted party and includes a encryption function, the trusted device being arranged to generate a response to a received challenge, the response comprising an acquired integrity metric and the authenticated integrity metric, both signed by the encryption function using the private asymmetric encryption key.

11. A trusted device configured for use in computing apparatus according to any one of the preceding claims.

12. A method of operating a system comprising trusted computing apparatus and a user, the trusted computing apparatus incorporating a trusted device being arranged to acquire the true value of an integrity metric of the computing apparatus, the method comprising the steps of:

the trusted device acquiring the true value of the integrity metric of the trusted computing apparatus;

the user generating a challenge for the trusted computing apparatus to prove its integrity and submitting the challenge to the trusted computing apparatus;

the trusted computing apparatus receiving the challenge, and the trusted device generating a response including the integrity metric and returning the response to the user; and

the user receiving the response, extracting the integrity metric from the response and comparing the integrity metric with an authenticated metric for the trusted computing apparatus that had been generated by a trusted party.

13. A method according to claim 12, wherein the challenge includes a nonce, the response includes the integrity metric and the nonce, both digitally signed by the trusted device using a encryption algorithm, and the user verifies the integrity metric and the nonce using a respective encryption algorithm.

14. A method according to claim 13, wherein the trusted device uses a private encryption key to sign the integrity metric and the nonce, and the user uses the respective public encryption key to verify the integrity metric and the nonce.

15. A method according to claim 14, wherein the response includes a certificate held by the trusted device, which certificate has been digitally signed by a trusted party using a private encryption key of the trusted party, the certificate including the public encryption key of the trusted device, and the user verifies the certificate using the public encryption key of the trusted party and uses the public encryption key from the certificate to verify the integrity metric and the nonce.

16. A method of establishing a communications channel in a system between trusted computing apparatus and remote computing apparatus, the method including the step of the remote computing apparatus verifying the integrity of the trusted computing apparatus using the method according to any one of claims 12 to 15, and maintaining the communications channel for further transactions in the event the integrity of the trusted computing apparatus is successfully verified by the remote computing apparatus.

17. A method of verifying that trusted computing apparatus is trustworthy for use by a user for processing a particular application, the method including the step of the user verifying the integrity of the trusted computing apparatus using the method according to any one of claims 12 to 15, and the user using the trusted computing apparatus to process the particular application in the event the integrity of the trusted computing apparatus is successfully verified by the remote computing apparatus.

18. Trusted computing apparatus adapted for use in accordance with the method of any one of claims 12 to 17.

19. Remote computing apparatus arranged for use in accordance with claim 16.

20. A trusted device arranged for use in accordance with any one of claims 12 to 17.

21. Computing apparatus configured to receive a trusted device as claimed in claim 11.

ABSTRACT

Trusted Hardware Device in a Computing Platform

In a computing platform, a trusted hardware device (14) is added to the motherboard (10). The trusted hardware device (14) is configured to acquire an integrity metric, for example a hash of the BIOS memory (19), of the computing platform. The trusted hardware device (14) is tamper-resistant, difficult to forge and inaccessible to other functions of the platform. The hash can be used to convince users that that the operation of the platform (hardware or software) has not been subverted in some way, and is safe to interact with in local or remote applications.

In more detail, the main processing unit (11) of the computing platform is directed to address the trusted hardware device (14), in advance of the BIOS memory, after release from 'reset'. The trusted hardware device (14) is configured to receive memory read signals from the main processing unit (11) and, in response, return instructions, in the native language of the main processing unit (11), that instruct the main processing unit to establish the hash and return the value to be stored by the trusted hardware device (14). Since the hash is calculated in advance of any other system operations, this is a relatively strong method of verifying the integrity of the system. Once the hash has been returned, the final instruction calls the BIOS program and the system boot procedure continues as normal.

Whenever a user wishes to interact with the computing platform, he first requests the integrity metric, which he compares with an authentic integrity metric that was measured by a trusted party. If the metrics are the same, the platform is verified and interactions can continue. Otherwise, interaction halts on the basis that the operation of the platform may have been subverted.

Figure 5.

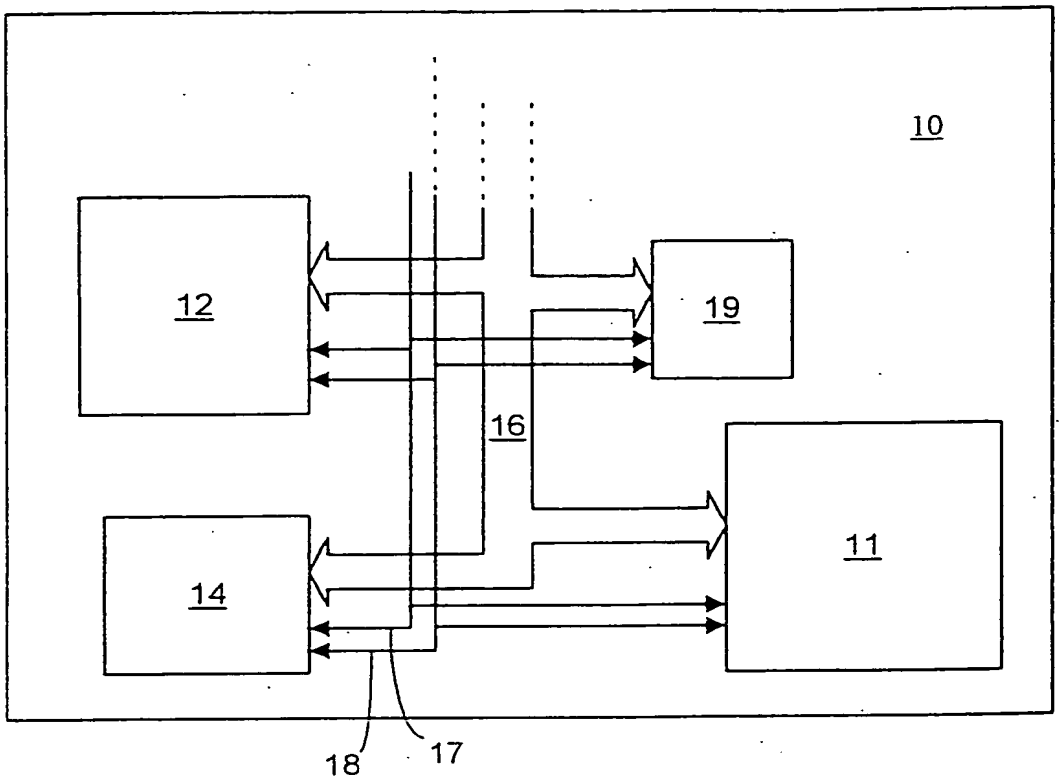


FIGURE 1

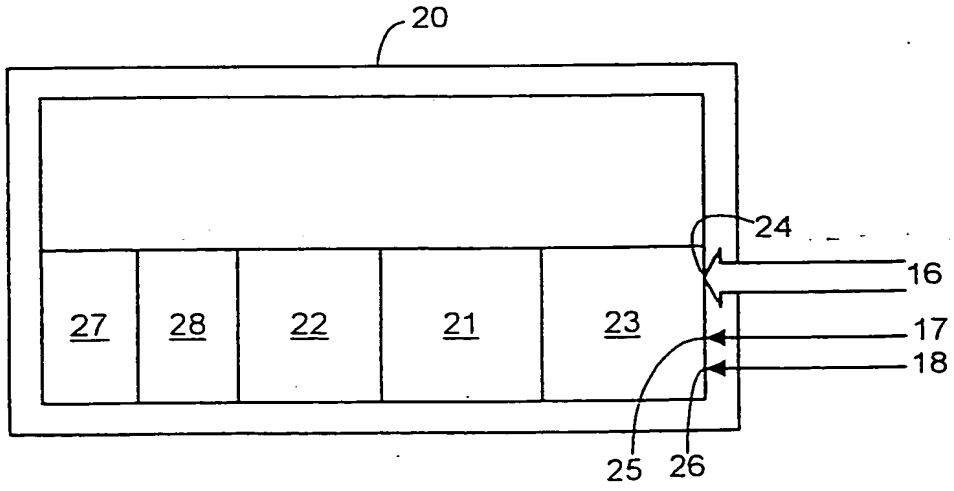


FIGURE 2

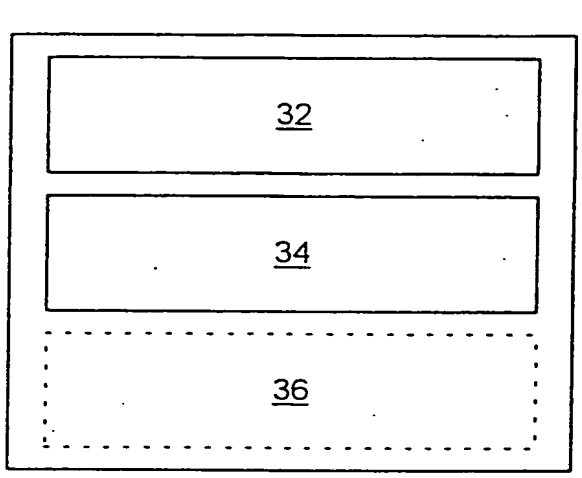


FIGURE 3

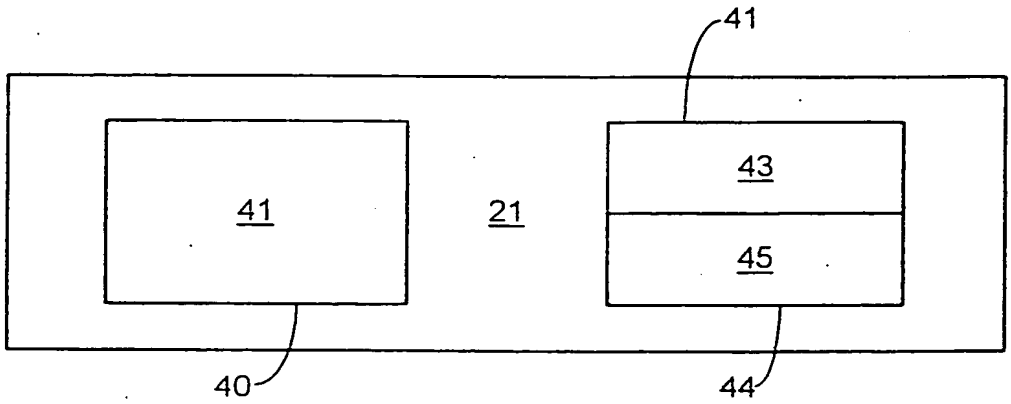


FIGURE 4

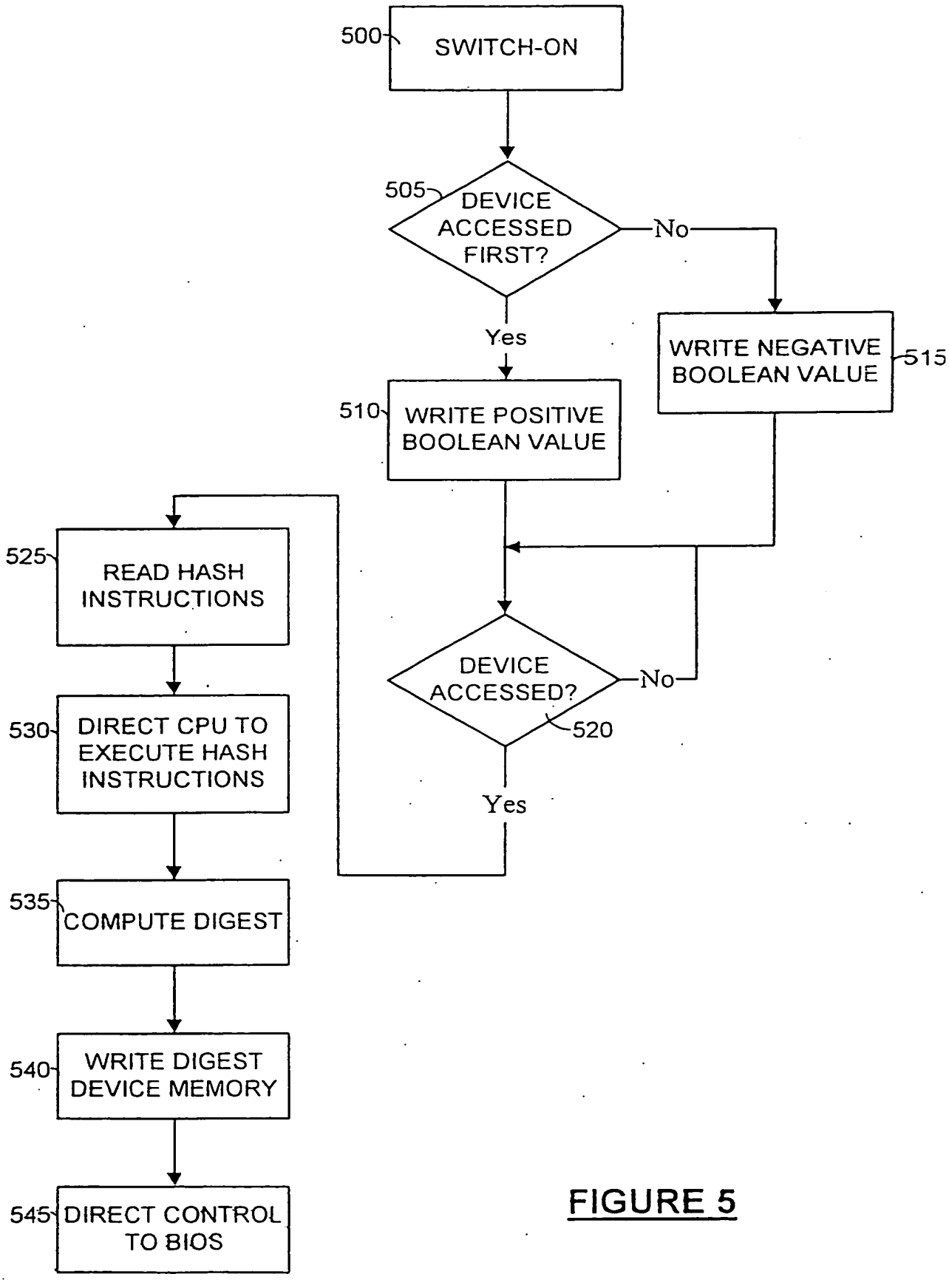

500
SWITCH-ON
505
DEVICE ACCESSED FIRST?
No
Yes
WRITE NEGATIVE BOOLEAN VALUE
515
510
WRITE POSITIVE BOOLEAN VALUE
525
READ HASH INSTRUCTIONS
530
DIRECT CPU TO EXECUTE HASH INSTRUCTIONS
DEVICE ACCESSED?
No
520
Yes
535
COMPUTE DIGEST
540
WRITE DIGEST DEVICE MEMORY
545
DIRECT CONTROL TO BIOS

FIGURE 5

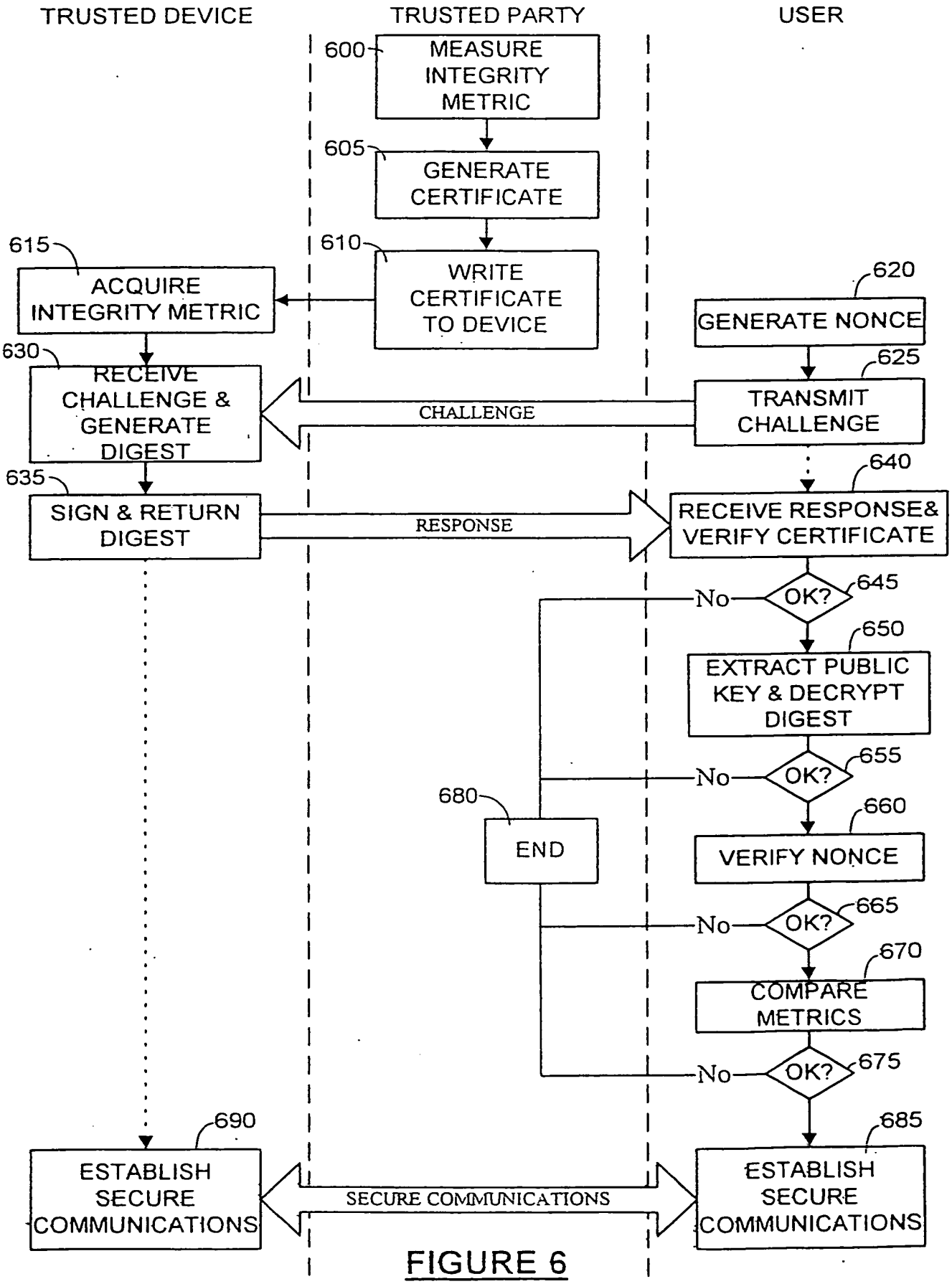

TRUSTED DEVICE
TRUSTED PARTY
USER
600
MEASURE INTEGRITY METRIC
605
GENERATE CERTIFICATE
610
WRITE CERTIFICATE TO DEVICE
615
ACQUIRE INTEGRITY METRIC
620
GENERATE NONCE
625
TRANSMIT CHALLENGE
CHALLENGE
630
RECEIVE CHALLENGE & GENERATE DIGEST
635
SIGN & RETURN DIGEST
RESPONSE
640
RECEIVE RESPONSE& VERIFY CERTIFICATE
No
OK?
645
650
EXTRACT PUBLIC KEY & DECRYPT DIGEST
No
OK?
655
680
END
660
VERIFY NONCE
No
OK?
665
670
COMPARE METRICS
No
OK?
675
690
ESTABLISH SECURE COMMUNICATIONS
SECURE COMMUNICATIONS
685
ESTABLISH SECURE COMMUNICATIONS

FIGURE 6